Primitive types

Little endian format.

Type	Length	Description
Float	4	
Int32	4	Signed
UInt32	4	Unsigned
UInt64	8	Unsigned
Bool	**4**	True if 1, else false
Int16	2	Signed
UInt16	2	Unsigned
ChunkName	4	RIFF-style chunk identifier
UTF8String		Null-terminated

Enum types

```
SoundEntryFlags : UInt32
 ├─IsEmbedded = 1
 ├─IsCompressed = 2
 └─Regular = 0x64 // everything seems to have these flags set
RoomEntryFlags : UInt32
 ├─EnableViews = 1
 ├─ShowColor = 2
 └─ClearDisplayBuffer = 4
CollisionShape : UInt32
 ├─Circle = 0
 ├─Box = 1
 └─Custom = 2
InfoFlags : UInt32
 ├─Fullscreen = 0x0001
 ├─SyncVertex1 = 0x0002
 │
```

```
├SyncVertex2 = 0x0004
├Interpolate = 0x0008
├unknown = 0x0010
├ShowCursor = 0x0020
├Sizeable = 0x0040
├ScreenKey = 0x0080
├SyncVertex3 = 0x0100
├StudioVersionB1 = 0x0200
├StudioVersionB2 = 0x0400
├StudioVersionB3 = 0x0800
├StudioVersionMask = 0x0E00 // studioVersion = (infoFlags & InfoFlags.StudioVersionMask) >> 9
├SteamEnabled = 0x1000
├LocalDataEnabled = 0x2000
└BorderlessWindow = 0x4000
GameTargets : UInt32
// not discovered yet...
```

Complex types

```
String
└StringAddress : Int32 //The value is an UTF8String
List<T>
├AddressCount : Int32
├Addresses : Int32[AddressCount]
└T[Addresses.Length]
Chunk
├Name : ChunkName
└Length : Int32
ListChunk<T> : Chunk
└List<T>
RefDefEListChunk : Chunk Only valid for bytecode version 0xE (see FUNC/VARI for 0xF)
└RefDefEList : RefDefE[] //Read until end; logic explained in decompilation process
  ├Name : String
  ├Occurrences : Int32
  └FirstAddress : Int32
Form : Chunk
├Gen8 : Chunk //Metadata
│ ├Debug : Byte
│ ├unknown : Int24
│ ├Filename : String
│ ├Config : String
│ ├LastObj : UInt32 // possibly the last offset of all objects, but this is untested
│ ├LastTile : UInt32 // idem
│ ├GameID : UInt32
│ ├unknown : UInt32[4]
│ ├Name : String
│ ├Major : Int32
│ │
│   ├─Minor : Int32
│   ├─Release : Int32
│   ├─Build : Int32
│   ├─DefaultWindowWidth : Int32
│   ├─DefaultWindowHeight : Int32
│   ├─Info : InfoFlags
│   ├─LicenseMD5 : Byte[0x10]
│   ├─LicenseCRC32 : UInt32
│   ├─Timestamp : UInt64 // UNIX time
│   ├─DisplayName : String
│   ├─ActiveTargets : GameTargets // probably flags indicating for which platforms the file is built, but no flag values
│   │ are known at this point
│   ├─unknown : UInt32[4]
│   ├─SteamAppID : UInt32
│   ├─NumberCount : UInt32
│   └─Numbers : UInt32[NumberCount]
├─Optn : Chunk
│   ├─unknown : UInt32[2]
│   ├─Info : InfoFlags // duplicate from GEN8
│   ├─unknown : UInt32[0xC]
│   └─ConstantMap : List<Constant>
│       ├─Name : String
│       └─Value : String
├─Extn : Chunk //Empty // NOTE: a rough structure is known, but too vague to include here. Read the Altar.NET src
├─Sond : Chunk //Sound data
│   ├─Name : String
│   ├─Flags : SoundEntryFlags
│   ├─Type : String
│   ├─File : String
│   ├─unknown : UInt32
│   ├─Volume : Float
│   ├─Pitch : Float
│   ├─GroupID : Int32 // to AGRP
│   └─AudioID : Int32 // actual audio data, -1 when not embedded
├─Agrp : ListChunk<AudioGroup>
│   └─Name : String
├─Sprt : ListChunk<Sprite>
│   ├─Name : String
│   ├─Width : Int32
│   ├─Height : Int32
│   ├─MarginLeft : Int32
│   ├─MarginRight : Int32
│   ├─MarginBottom : Int32
│   ├─MarginTop : Int32
│   ├─unknown : UInt32[3] // maybe something with collision masks
│   ├─BBoxMode : UInt32
│   ├─SepMasks : UInt32
│   ├─OriginX : UInt32
│   ├─OriginY : UInt32
│   │
│  ├─TextureCount : Int32
│  ├─TextureAddresses : Int32[TextureCount] // to TPAG
│  └─Unknown : Byte[] //Until next object
├─Bgnd : ListChunk<Background>
│  ├─Name : String
│  ├─unknown : UInt32[3]
│  └─TextureAddress : Int32 // to TPAG
├─Path : ListChunk<Path> //Paths
│  ├─Name : String
│  ├─IsSmooth : Bool
│  ├─IsClosed : Bool
│  ├─Precision : UInt32
│  └─Points : List<UInt32>
│     ├─X : Float
│     ├─Y : Float
│     └─Speed : Float
├─Scpt : ListChunk<ScriptDefinition>
│  ├─Name : String
│  └─Id : UInt32 // to CODE
├─Shdr : Chunk //Empty
├─Font : Chunk //Fonts
│  ├─CodeName : String
│  ├─SystemName : String
│  ├─EmSize : UInt32
│  ├─Bold : Bool
│  ├─Italic : Bool
│  ├─RangeStart : UInt16 // ignore this, use the character list instead
│  ├─Charset : Byte
│  ├─AntiAliasing : Byte
│  ├─RangeEnd : UInt32
│  ├─TPagId : UInt32 // TPAG containing the glyphs
│  ├─ScaleX : Float
│  ├─ScaleY : Float
│  └─Glyphs : List<Glyph>
│     ├─Character : UInt16 // 16-bit codepoint
│     ├─RelativeX : UInt16
│     ├─RelativeY : UInt16
│     └─unknown : Byte[6]
├─Tmln : Chunk //Empty
├─Objt : ListChunk<GameObjectDefinition>
│  ├─Name : String
│  ├─SpriteIndex : Int32
│  ├─Visible : Bool
│  ├─Solid : Bool
│  ├─Depth : Int32
│  ├─Persistent : Bool
│  ├─ParentId : Int32 // -1 if none
│  ├─TextureMaskId : Int32 // -1 if none
│  ├─UsesPhysics : Bool
│  │
```

```
 │  ├ IsSensor : Bool
 │  ├ CollisionShape : CollisionShape
 │  ├ Physics : ObjectPhysics
 │  │  ├ Density : Float
 │  │  ├ Restitution : Float
 │  │  ├ Group : Float
 │  │  ├ LinearDamping : Float
 │  │  ├ AngularDamping : Float
 │  │  ├ unknown : Float
 │  │  ├ Friction : Float
 │  │  ├ unknown : Float
 │  │  └ Kinematic : Float
 │  │ // NOTE: sometimes, more floats are here as well, the exact conditions are unknown. See the Altar.NET source.
 │  ├ ShapePointCount : UInt32
 │  └ ShapePointOffsets : UInt32 // read the Altar.NET source
 ├ Room : ListChunk<Room>
 │  ├ Name : String
 │  ├ Caption : String
 │  ├ Width : UInt32
 │  ├ Height : UInt32
 │  ├ Speed : UInt32
 │  ├ Persistent : Bool
 │  ├ Argb : UInt32
 │  ├ DrawBGColor : Bool
 │  ├ unknown : UInt32
 │  ├ Flags : RoomEntryFlags
 │  ├ BgOffset : UInt32 // offsets to the List<T> later on
 │  ├ ViewOffset : UInt32
 │  ├ ObjOffset : UInt32
 │  ├ TileOffset : UInt32
 │  ├ World : UInt32
 │  ├ Top : UInt32
 │  ├ Left : UInt32
 │  ├ Right : UInt32
 │  ├ Bottom : UInt32
 │  ├ GravityX : Float
 │  ├ GravityY : Float
 │  ├ MetresPerPixel : Float
 │  ├ Backgrounds : List<Background>
 │  │  ├ Enabled : Bool
 │  │  ├ Foreground : Bool
 │  │  ├ BgDefIndex : UInt32
 │  │  ├ X : UInt32
 │  │  ├ Y : UInt32
 │  │  ├ TileX : Bool
 │  │  ├ TileY : Bool
 │  │  ├ SpeedX : UInt32
 │  │  ├ SpeedX : UInt32
 │  │  └ ObjectId : Int32
 │  │
```

```
│  ├─Views : List<View>
│  │  ├─Enabled : Bool
│  │  ├─ViewX : Int32
│  │  ├─ViewY : Int32
│  │  ├─ViewWidth : Int32
│  │  ├─ViewHeight : Int32
│  │  ├─PortX : Int32
│  │  ├─PortY : Int32
│  │  ├─PortWidth : Int32
│  │  ├─PortHeight : Int32
│  │  ├─BorderX : UInt32
│  │  ├─BorderY : UInt32
│  │  ├─SpeedX : UInt32
│  │  ├─SpeedY : UInt32
│  │  └─ObjectId : Int32
│  ├─GameObjects : List<GameObject>
│  │  ├─X : Int32
│  │  ├─Y : Int32
│  │  ├─BgDefIndex : Int32
│  │  ├─InstanceID : Int32
│  │  ├─CreationCodeID : Int32 // to CODE (-1 for none) -> gml_RoomCC_<name>_<CreationCodeID>
│  │  ├─ScaleX : Float
│  │  ├─ScaleY : Float
│  │  ├─ARGBTint : UInt32
│  │  └─Rotation : Float
│  └─Tiles : List<Tile>
│     ├─X : Int32
│     ├─Y : Int32
│     ├─BgDefIndex : Int32
│     ├─SourceX : Int32
│     ├─SourceY : Int32
│     ├─Width : UInt32
│     ├─Height : UInt32
│     ├─TileDepth : Int32
│     ├─InstanceID : Int32
│     ├─ScaleX : Float
│     ├─ScaleY : Float
│     └─ARGBTint : UInt32
├─Dafl : Chunk //Empty
├─Tpag : ListChunk<Texture>
│  ├─X : UInt16
│  ├─Y : UInt16
│  ├─Width : UInt16
│  ├─Height : UInt16
│  ├─RenderX : UInt16
│  ├─RenderY : UInt16
│  ├─BoundingX : UInt16
│  ├─BoundingY : UInt16
│  ├─BoundingWidth : UInt16
│  │
│  ├─BoundingHeight : UInt16
│  └─SpritesheetId : UInt16
├─Code : ListChunk<CodeE> // bytecode version 0xE
│  ├─Name : String
│  ├─Length : UInt32
│  └─Code : Byte[Length] // or until next object
├─Code : ListChunk<CodeF> // bytecode version 0xF
│  ├─Name : String
│  ├─Length : UInt32
│  ├─unknown : UInt32
│  ├─BytecodeAddress : Int32 // offset to the actual bytecode, relative to this value
│  └─unknown : UInt32
├─Vari : RefDefEListChunk // if bytecode version == 0xE
├─Vari : ListChunk&lit;VariableDefinition> // if bytecode version == 0xF
│  ├─Name : String
│  ├─unknown : UInt32[2]
│  ├─Occurrences : UInt32
│  └─FirstAddress : UInt32
├─Func : RefDefEListChunk // for both bytecode versions, it seems
│ // For detailed information on RefDef parsing, read the Altar.NET source
├─Strg : ListChunk<StringDefinition>
│  ├─Length : UInt32
│  └─Value : UTF8String
├─Txtr : ListChunk<Spritesheet>
│  ├─unknown : UInt32
│  └─PngAddress : UInt32
└─Audo : ListChunk<Audio>
   ├─Length : UInt32
   └─WavBlob : Byte[Length] // or was it Length + 4?
```